CLUSTER FINDING

MATT ZHANG

Cluster Finding

- Problem recap - given an event with multiple objects (such as two jets), we want to be able to “zoom in” and cut out a section of the ECAL and HCAL around each object.

- Three methods of doing this:
  - Pass truth 4-vectors to python script for creating ECAL and HCAL sections.
  - Use PandoraPFA package to find and return clusters from ROOT files, then use python script to cut out ECAL and HCAL sections.
  - Modify python script to directly find clusters in ROOT files, as opposed to just calculating the barycenter.

Truth 4-Vectors

```
HepMC::Version 2.06.08
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 1.9791216363684406e+01 1.7157824146590872e-01 7.6844392989166835e-03 211 0 7 1 2 0 1 1.0000000000000000e-00
N 1 "0"
U GEV MM
C 2.1474891241584755e+04 2.1474891241584755e+04
F 11 -11 9.9999999849621621e-01 9.9999999967730668e-01 1.9791216363684406e+01 4.5124267539581425e+07 1.9567989008858083e+08 0 0
V -1 0 0 0 0 0 0 1 0
P 3 11 0 0 2.4999999962406156e+02 2.4999999962406156e+02 0 21 0 0 -3 0
V -2 0 0 0 0 0 0 1 0
P 4 -11 0 0 -2.4999999991932668e+02 2.4999999991932668e+02 0 21 0 0 -3 0
V -3 0 0 0 0 0 0 2 0
P 5 11 -5.4138532577453384e+00 -1.9036345186409278e+01 2.4921538388881373e+02 2.4999999962452483e+
P 6 -11 5.4138532577453384e+00 1.9036345186409278e+01 -2.4921538418407886e+02 2.4999999991886335e+
V -4 0 0 0 0 0 1 2 0
P 1 11 0 0 2.4999999999947775e+02 2.4999999999999937e+02 5.1099999999999995e-04 4 0 0 -4 0
P 7 11 0 0 2.4999999962406156e+02 2.4999999962406156e+02 0 61 0 0 -1 0
P 11 22 0 0 3.7593844176604381e-07 3.7593844176604381e-07 0 1 0 0 0 0
V -5 0 0 0 0 0 1 2 0
P 2 -11 0 0 -2.4999999999947775e+02 2.4999999999999937e+02 5.1099999999999995e-04 1 0 0 -5 0
P 8 -11 0 0 -2.4999999991932668e+02 2.4999999991932668e+02 0 61 0 0 -2 0
P 12 22 0 0 -8.0673316915635938e-08 8.0673316915635938e-08 0 1 0 0 0 0
V -6 0 0 0 0 0 0 1 0
P 9 11 -5.4138532577453384e+00 -1.9036345186409278e+01 2.4921538388881373e+02 2.4999999962452483e+
V -7 0 0 0 0 0 0 1 0
P 10 -11 5.4138532577453384e-00 1.9036345186409278e+01 -2.4921538418407886e+02 2.4999999991886335e+
E 1 -1 3.5806279837882848e+00 2.6946965035197523e-01 7.5435569404292951e-03 211 0 12 1 2 0 1 1.000
N 1 "0"
U GEV MM
C 1.0342162216312715e+05 6.3898810583215680e+04
```

V - GenVertex information

- **int**: *barcode*
- **int**: *id*
- **double**: x
- **double**: y
- **double**: z
- **double**: *ctau*
- **int**: *number of "orphan" incoming particles*
- **int**: *number of outgoing particles*
- **int**: *number of entries in weight list (may be zero)*
- **double**: *optional list of weights*

P - GenParticle information

- **int**: *barcode*
- **int**: *PDG id*
- **double**: *px*
- **double**: *py*
- **double**: *pz*
- **double**: *energy*
- **double**: *generated mass*
- **int**: *status code*
- **double**: *Polarization theta*
- **double**: *Polarization phi*
- **int**: *barcode for vertex that has this particle as an incoming particle*
- **int**: *number of entries in flow list (may be zero)*
- **int, int**: *optional code_index and code for each entry in the flow list*

Truth 4-Vectors

- Need to decide what to take as our truth 4-vectors here. Should we take the first two particles with PdgID corresponding to π0 and use their (px, py, pz, m) vector?

- Working on a script to take a HepMC file and extract the relevant 4-vectors in a text file (after which the file can be used by the python script).

- Need to know about detector geometry - given an eta and phi, at what (x, y, z) will the calorimeter be hit?

- Deciding which vectors to take as truth 4-vector is process dependent. This should not be a big deal if we are only generating a few types of processes.

Pandora

- Another option is to use the DDMarlinPandora package.

- This can take the output of Marlin and send it to PandoraPFA, which outputs things like cluster positions.

- DDMarlinPandora is an offshoot of the MarlinPandora package and appears to be in development - it may be a bit difficult to use right now.

Custom Algorithm

- We have the entire ECAL and HCAL information. Based on this we can create a simple clustering algorithm in python.

- Start with a grid of values → find the pixel with the highest value.
    - Look in a radius of two pixels around that pixel, and for all pixels which have a lesser value, but still above the threshold, add it to the cluster.
    - Continue doing this until the cluster no longer grows.
    - Repeat process with highest-value remaining pixel.

- This approach would allow us to get variable-sized chunks of ECAL and HCAL data (containing the entire cluster), rather than a uniform sized slice.